XML Technology Enables Redesigned Deep Space Network Operations[1][2]

Ruth Bergman
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91009
818-354-9116
ruth@jpl.nasa.gov

Abstract—NASA's Deep Space Network (DSN) provides telecommunication support for all NASA missions. Successful support relies on a great deal of information about each mission from equipment type to trajectory and encounter times. In the past, with only a handful of missions to support, this information was collected manually and telecommunication services were agreed upon manually. In the future, with an order of magnitude more missions flying, the DSN must automate its processes.

The barriers to automation in the DSN environment are many: heterogeneous computing environments, legacy systems, existing procedures, etc. We advocate the use of the Extensible Markup Language (XML) as an enabling technology in our quest to automate DSN operations. In our development we are leveraging XML technology to solve several information exchange problems. This paper discusses three applications of XML to DSN automation: (1) the service request process, (2) legacy application integration, and (3) a DSN portal.

TABLE OF CONTENTS

1. INTRODUCTION

The new breed of NASA missions, with more missions at less cost, requires JPL's Telecommunications and Mission Operations Directorate (TMOD) to review and redesign the operational concepts of NASA's Deep Space Network (DSN) - the network that provides communication with spacecrafts in flight. The new operational concept for the DSN emphasizes standard mission services and distributed control of station operations. Furthermore, the operational concept makes use of well-defined, user-friendly, automated interfaces among TMOD, the DSN stations, and missions throughout the lifecycle of the mission from planning through operations. The challenges we face in automating the interfaces for DSN operations revolve around the arising need for fast, automated data-exchange. The barriers to this goal are very likely familiar to all those in the aerospace industry.

- Heterogeneous operating systems
- Incompatible applications and tools
- Legacy data and applications
- Critical data that exists only on paper with no electronic version
- Need for real-time information dissemination
- Each player needs to view partial data in different formats
- A culture that is reluctant to abandon existing formats and procedures

Our solution to the problem includes bringing all the data and applications into a web-based client/server environment and using the Extensible Markup Language (XML) to enable data exchange.

XML is a new recommended standard for Markup Languages from the World Wide Web Consortium (W3C) [2]. XML extends the capabilities of HTML, which uses presentation specific markup tags. XML, by comparison, emphasizes content, not presentation. Users define their own tags. Presentation is separate from the content and the user can define the presentation for user-defined tags. XML is hailed as the technology that will revolutionize the web [1] and is quickly gaining popularity by a wide range of Internet users and companies. XML arose from the world of document authoring and maintenance, but it is becoming the standard for information exchange on the Internet.

Naturally, there are many opportunities in aerospace for XML as a document authoring and content management system. These problems, however, are mundane in comparison with the crux of aerospace missions and

[1] 0-7803-5846-5/00/$10.00 © 2000 IEEE
[2] This work was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

operations. This paper is concerned with our concept for leveraging XML information exchange capabilities to solve engineering problems.

This paper will illustrate the usefulness of XML to aerospace applications with examples from our development for the DSN. To support the new operational concept for the DSN we are using the Internet and XML technology to enable automation of interfaces for both DSN/mission and DSN/DSN interaction. We are looking to XML to solve the following problems:
(1) Service Request Process. At present the process of requesting DSN services is very ad-hoc and labor intensive. The interfaces for this process are not well defined, with data often exchanged on paper or email. Defining interfaces for this process using a Service Request Language derived from XML is an inexpensive solution to enable automated data-exchange for the service request process.
(2) Application Integration. At JPL, like all highly specialized companies, there are many legacy applications that represent decades of development and expertise. These applications typically run on a Unix machine as FORTRAN or C programs and are costly to re-implement in an Internet-based solution. XML, because it is a text format, enables a straightforward solution to the problem for application integration. We have prototyped an application integration solution using XML-RPC, a remote procedure call protocol in XML [3].
(3) DSN Operational Portal. We are advocating the use of a portal for disseminating real-time operational information needed for DSN operations. DSN operators around the world as well as flight project personnel and possibly program managers provide information to the portal and have access to the information the portal provides. Through the Internet and portal publish/subscribe and "information push" capabilities newly published information is available to all subscribers at regular (nearly immediate) updates in a secure manner. This information will manifest itself to the user, who simply connects to the portal.

The remainder of this paper is organized as follows. Section 2 discusses the functionality and capability of the Deep Space Network, and difficulties operating this complex network present. Section 3 introduces the Extensible Markup Language. Section 4 describes the three specific problems we intend to solve via XML, and the status of our development of these solutions. Finally, section 5 summarizes our findings.

2. The Deep Space Network (DSN)

The NASA Deep Space Network is a world-wide network of antennas with the principal function of supporting interplanetary spacecraft missions for the exploration of the solar system and the universe. The DSN consists of three tracking Complexes located 120 degrees apart around the world: in Goldstone, California (USA), near Canberra, (Australia) and Madrid (Spain). The DSN was built and is managed by the Jet Propulsion Laboratory of the California Institute of Technology. Within JPL, responsibility for managing the DSN is assigned to the Telecommunications and Mission Operations Directorate (TMOD).

Space exploration is accomplished by unmanned, automated spacecraft. The DSN antennas provide two-way communication with the spacecraft. The DSN's antennas and data-delivery system make it possible to:
- Acquire telemetry data from spacecraft.
- Transmit commands to spacecraft.
- Track spacecraft position and velocity.
- Perform very-long-baseline interferometry observations.
- Measure variations in radio waves for radio science experiments.
- Gather science data.
- Monitor and control the performance of the network.

For more information about the DSN see [4].

JPL's unmanned missions in the solar system are dependent upon the services of the DSN. Without these services space exploration would not be possible. Thus, every mission from its very beginning must consider its telecommunication needs. Throughout mission design, development and operation the mission personnel must be in contact with DSN personnel to ensure that the necessary services will be provided. Early in mission planning telecommunication feasibility is assessed. The designer needs to know if there is an antenna that will be able to communicate with the spacecraft at a particular time and position in space and whether the communication will have sufficient data rate. Later in the mission planning the mission and DSN agree on a set of "tracks", with specific antennas committed to a chunk of time for this mission. In the last few weeks before an event specific equipment for use with the antenna must be allocated. Up to now, the process of defining services, tracks, scheduling antennas and scheduling equipment has been very ad hoc and manually intensive. Track schedules are negotiated in meetings including representatives of every mission and the DSN. Information concerning services including any corrections, changes and problems are transmitted via e-mail, phone, on paper, etc. It is this operational concept that we wish to revise and automate.

3. The Extensible Markup Language (XML)

XML is a language for defining text-based markup. Most readers are probably familiar with text markup in HTML. Unlike HTML whose predefined markup tags are presentation specific, XML tags are user-defined and carry no presentation related information. Whereas HTML is a specific language, XML can be viewed as a meta-language.

From a historical perspective XML is derived from the Standard Generalized Markup Language (SGML). SGML has been the markup language of choice for creating and maintaining large documents. The drawback of SGML is its complexity. Web developers have long seen a need for more flexible markup than HTML provides. SGML provides the needed capabilities, but SGML and SGML tools are too simply too difficult to learn for typical web development. Thus, a markup language that provides the flexibility and extensibility of SGML with the ease-of-use of HTML was needed. The World Wide Web Consortium (W3C) took up the task of creating this language and XML is the result of this development.

The following "hello world" example illustrates simple use of XML. A very simple XML example:

```
<GREETING>Hello World!</GREETING>
```

If you save the above in a file you can view it in any XML enabled application such as Internet Explorer 5. Note that the tag name GREETING is not an HTML tag, nor does it carry any presentation information. The user can define the presentation for the tags in the document using a stylesheet language such as the Extensible Stylesheet Language (XSL) which is also a W3C development. XSL is also an application of XML. That is, the XSL language is defined in XML markup. The following is a sample portion of an XSL stylesheet for the hello world example. This portion shows how the stylesheet matches elements of type GREETING and applies presentation to the content. (Appendix A contains the complete XSL stylesheet for this example.)

```
<xsl:template match="GREETING">
  <DIV STYLE="font-family:Arial; font-size:15pt;
                    color:red;" ...
  </DIV>
</xsl:template>
```

The separation of the presentation from the content of XML documents is one of its most useful features. First, tags can carry meaningful names such as <GREETING> or <NAME> instead of HTML's generic <TITLE> or <P>. These tags allow flexible access to the content of the document, thus enabling flexible web applications, in particular, more meaningful searches. Second, a document can have multiple presentation styles, for example, a plain text style and an HTML style. Likewise, multiple documents can have the same style. Lastly, the content and presentation of the document can be updated separately to enable easier document maintenance.

XML provides built-in validation through a Document Type Definition (DTD). The user defines the proper nesting of elements in a DTD. For example the following example is a DTD for contact information.

```
<!ELEMENT PERSON (NAME, CONTACTINFO+)>
<!ELEMENT NAME (PREFIX*,FIRSTNAME,
LASTNAME)>
<!ELEMENT PREFIX (#PCDATA)>
<!ELEMENT FIRSTNAME (#PCDATA)>
<!ELEMENT LASTNAME (#PCDATA)>
<!ELEMENT CONTACTINFO (ADDRESS, PHONE,
EMAIL,(FAX)*)>
<!ATTLIST CONTACTINFO LOCATION (work|home)
"home">
<!ELEMENT ADDRESS (STREETADDRESS+, CITY,
STATE, COUNTRY*, ZIPCODE)>
<!ELEMENT STREETADDRESS (#PCDATA)>
<!ELEMENT CITY (#PCDATA)>
<!ELEMENT STATE (#PCDATA)>
<!ELEMENT COUNTRY (#PCDATA)>
<!ELEMENT ZIPCODE (#PCDATA)>
<!ELEMENT PHONE (#PCDATA)>
<!ELEMENT EMAIL (#PCDATA)>
<!ELEMENT FAX (#PCDATA)>
```

The following document provides the information for one contact while adhering to the requirements in the DTD. Note that the document refers to the document type using the DOCTYPE statement.

```
<?xml version="1.0"?>
<!DOCTYPE RESUME SYSTEM "resume.dtd" >
<PERSON>
  <NAME>
    <PREFIX>Dr.</PREFIX>
    <FIRSTNAME>Ruth</FIRSTNAME>
    <LASTNAME>Bergman</LASTNAME>
  </NAME>
  <CONTACTINFO LOCATION="work">
    <ADDRESS>
      <STREETADDRESS>Mail Stop 301-170V
      </STREETADDRESS>
      <STREETADDRESS>4800 Oak Grove Drive
      </STREETADDRESS>
     <CITY>Pasadena</CITY>
     <STATE>CA</STATE>
     <ZIPCODE>91109</ZIPCODE>
    </ADDRESS>
    <PHONE>(818) 354-9116</PHONE>
    <EMAIL>ruth@jpl.nasa.gov</EMAIL>
  </CONTACTINFO>
</PERSON>
```

The advantage of using XML, because it is an open standard, as opposed to a proprietary format is that commercial tools support this format. One can download a number of parsers that comply to the XML standard and provide the built-in DTD validation. Using an off-the-shelf tool is a significant advantage for an application developer. It implies faster application development and easier maintenance. Similarly, there are advantages for the user of the XML language or application. For editing and viewing the user can choose from many XML editors. Furthermore, because XML is a text based format it can be edited and viewed in any text editor (although this may not be the most easy to read format) and on any computing platform.

So far we have assessed XML primarily as a document creation and management technology. Because of its historical origin from SGML, XML is often erroneously perceived as a document oriented technology. It is by now clear, that XML will have a tremendous role in enabling data exchange, in particular on the Internet. XML is already used by many electronic commerce applications to replace proprietary format for data exchange. Since XML is lightweight and text-based it easily lends itself to transfer over HTTP and on-the-fly creation and interpretation. Companies can leverage the existence of a standard and supporting tools to speed up application development and language definition.

In our design for the DSN we are considering XML for document creation and management of the many documents managed by the DSN. The more exciting applications of XML, however, are in the data exchange realm. These are the subject of the remainder of this paper.

4. Applications of XML for the DSN

Section 2 introduced the DSN and discussed the general problem of achieving reliable telecommunication in the complex system involving many spacecrafts and multiple antennas and antenna complexes. This section discusses some specific processes and related technology problems and illustrates how we use XML to solve them.

Service Request Process

To deliver communications services to the spacecraft, the DSN antenna operators require a great deal of information regarding the mission and spacecraft. Among these are

- *Communication hardware on the spacecraft* - such as antenna band and power.
- *Trajectory information* - location of the spacecraft at any time during its lifetime.
- *Antenna view-periods for the spacecraft* - the period during which the spacecraft is visible to the antenna. The spacecraft is usually visible only to one of the complexes at any time.
- *Spacecraft encounters* - mission critical encounters (such as an encounter with Mars) require very high, often constant, communication support.
- *Mission tracking requirements* - tracking needs vary between missions. Big missions may require several hours of tracking a day, whereas smaller missions may require a few hours of tracking a week. Important mission events can require continuous tracking for several days.

The mission must provide the spacecraft information to the DSN as well as tracking requirements. Actual tracking support is negotiated and agreed upon with the DSN.

The existing process of exchanging spacecraft information is completely manual. Mission representatives meet with DSN representatives to exchange information and negotiate tracking time. Furthermore, the process is ad-hoc, where the complete information set required to provide DSN services is nowhere stated concisely. While these processes have, thus far, been successful at supporting a handful of missions, they are too inefficient and costly to maintain in the future.

Our vision of the service request process is an almost-fully-automated, user-driven process. This process is supported by an integrated collection of computing tools and applications with a web-enabled, easy-to-use user interface. Starting with advanced mission planning the user can input a minimal number of mission parameters and within minutes receive critical data, namely, trajectory information, view-period information and track coverage forecasting. This information is vital to determine the viability of a proposed mission.

As the mission matures, more high-fidelity tools replace the baseline tools of advanced mission planning. Additional mission parameters are automatically requested by the applications as necessary for each phase. Track scheduling replaces track forecasting. Thus the services requested are refined over the mission planning cycle until a complete service package is produced. Human intervention should only be required in cases of contention over track scheduling.

There are many challenges in replacing the existing process with this vision:

1. Defining the information required for service requests. At present there is no definition for the set of parameters that comprise a complete service request. Furthermore, we want the flexibility of refining the service request over time. Thus we envision a family of languages, each with more detail about the spacecraft and services requested.
2. Defining a format or formats for service requests.
3. Developing the applications that enable the process. This is the subject of the next section on application integration.
4. Gaining acceptance of the user community. As is often the case with automation tools, the technological difficulties are outweighed by the cultural ones. How to get mission personnel and DSN operators to use the new process and tools? In our development, we follow recommended technology infusion guidelines, such as

prototyping, peer reviews and consulting with user groups.

To solve problems (1) and (2) above we advocate defining a Service Request Language in XML, or rather and XML compliant family of language. Each language will provide a viable description of the mission and requested service corresponding to the phase of the mission. When a mission enters a new phase and refinement of the service request is necessary, the new service request will build upon the previous phase service request in a natural manner.

Many features of XML lend to its use for a Service Request Language. It is text-based, thus a service request can be created in ordinary text editors as well as specialized editors. Furthermore, it can be used on any computing platform and easily transmitted using ftp or http protocols. We may find XML's built in international support useful as well. The separation of style from content is also beneficial for this problem. We will need multiple presentation styles for service requests. The missions will want a mission-oriented view: which antenna is supporting this mission at any time. Whereas, the DSN operators will want an antenna-oriented view: where is this antenna pointing at any time. In addition, we will likely have to support legacy formats for some years while people adapt to the new formats. Lastly, we can leverage on the popularity of XML when we build our service request related applications, by utilizing free or COTS software that knows how to deal with XML.

The following is an example of an XML service request. This example is purely for illustration and in no way consitutes an actual or complete service request. As you can see the request is for the Galileo mission, and several services are requested including command radiation, all frames and doppler services.

```
<?xml version="1.0"?>
<!DOCTYPE ServicePackage SYSTEM "srp.dtd" >
<ServicePackage>
  <ServiceGroup Name="Galileo_3_Station Series">
    <mission>gll</mission>

    <ServiceAllFrame Name="Service_1a">
      <downlink_band>S</downlink_band>
      <downlink_frequency units="MHz">2296.851852
      </downlink_frequency>
      <start_time>1996-122T01:20:00Z</start_time>
      <stop_time>1996-122T08:30:00Z</stop_time>
     </ServiceAllFrame>

    <ServiceCommandRadiation Name="Service_1b">
      <uplink_band>X</uplink_band>
      <uplink_frequency units="MHz">7168.091821
      </uplink_frequency>
      <start_time>1996-122T01:20:00Z</start_time>
      <stop_time>1996-122T08:30:00Z</stop_time>
    </ServiceCommandRadiation>

    <ServiceDoppler Name="Service_1c">
      <downlink_band>S</downlink_band>
      <uplink_band>X</uplink_band>
      <downlink_frequency units="MHz">2296.851852
      </downlink_frequency>
      <uplink_frequency units="MHz">7168.091821
      </uplink_frequency>
      <start_time>1996-122T01:20:00Z</start_time>
      <stop_time>1996-122T08:30:00Z</stop_time>
    </ServiceDoppler>
  </ServiceGroup>

</ServicePackage>
```

Much work is needed to define a fully specified Service Request Language. We believe that XML is the right technology to give us the necessary descriptive ability and flexibility for both the service requests documents and the people and applications that manipulate service requests.

Application Integration

JPL engineers and scientists use a variety of large, complex and critical applications. Applications that have been developed and improved over decades. These legacy applications are not realistically replaceable. It is simply to expensive to re-code and especially to re-validate these applications. This situation is probably typical of any large aerospace company.

For our development, it was clear from the beginning that integration is the only realistic solution. To make the problem worse, the legacy applications we need run on Unix machines, whereas our client/server application is a Windows NT application. There are several solutions to application communication using distributed object protocols. Microsoft provides effective solutions within the Windows environment, namely DCOM, but these solutions do not address other platforms. The CORBA standard provides solutions in heterogeneous environments. However, it involves a heavy installation and significant investment to learn. Furthermore, CORBA solutions only work with specific CORBA implementations. Most importantly, distributed object solutions do not work effectively in an environment with firewalls because they use dynamic port allocation. Since TMOD systems are in fact behind a firewall we prefer to avoid these problematic protocols.

The XML developer community offers a new option for interoperability - text-based remote procedure call transmitted over HTTP. A particular protocol of this type is the XML Remote Procedure Call protocol (XML-RPC)[?]. This XML conformant protocol leverages the benefits of XML, existing XML tools and HTTP to deliver an easy to implement remote procedure call mechanism. To assist developers further, the XML-RPC homepage lists several implementations of XML-RPC client and server code in a number of programming language. These modules handle the remote procedure invocation transparently. We use an XML-RPC client and server written by Hannes Wallofer [?] in a prototype we developed for the DSN.

We have implemented an Advanced Planning Prototype that, as the name indicates, provides advanced planning capability directly to the user on the web. To use this prototype the user provides a minimal set of mission and spacecraft parameters though a user-friendly GUI interface. The user can then directly invoke critical applications, such as trajectory generation, view-period generation, telecommunication capability and schedule forecasting. Thus, a mission planner can assess the feasibility of a mission in terms of DSN support very early in planning.

Some of the functionality of the Advanced Planning prototype was available to us in an existing legacy application. Specifically, trajectory generation and view-period generation applications that operate in the Unix environment exist. We wrapped these native applications with a Java class, which we provided to the XML-RPC Unix server. On the client side, the Windows NT server includes as an object another Java class that generates XML-RPC requests and transmits them to the Unix server and properly unwraps the XML-RPC result and returns the resulting values to the client. To the client the call to generate a view-period, for example, appears to be a regular method invocation.

The following samples contain a brief example of an XML-RPC request and reply. As you can see the XML request and reply are embedded in valid HTTP messages. The request calls uses the object vp and invokes the method genViewperiod. It has four parameters, an integer spacecraft id, the start time for the view-period, the end time, and a string trajectory file name.

```
POST /RPC2 HTTP/1.0
User-Agent: RuthPC (WinNT)
Host: slothrop.jpl.nasa.gov
Content-Type: text/xml
Content-length: 181


<?xml version="1.0"?>
<methodCall>
   <methodName>vp.genViewperiod
   </methodName>
   <params>
      <param>
         <value><i4>25</i4></value>
      </param>
      <param>
         <value><dateTime.iso8601>
         19990922T14:12:55
         </dateTime.iso8601></value>
      </param>
      <param>
         <value><dateTime.iso8601>
         20040717T14:00:03
         </dateTime.iso8601> </value>
       </param>
       <param>
         <value><string>SPK filename
         </string></value>
       </param>
    </params>
</methodCall>
```

The genViewperiod method returns a status value, in this case 0 indicating success.

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 158
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:08 GMT
Server: RuthPC-WinNT


<?xml version="1.0"?>
<methodResponse>
   <params>
      <param>
         <value>
            <int>0</int>
         </value>
      </param>
   </params>
</methodResponse>
```

XML-RPC provided a good solution for the Advanced Planning Prototype. There are other similar XML protocols. Most notably the Simple Object Access Protocol (SOAP) from Microsoft has been gaining support from the XML developer community [?]. Although at present, there is no standard protocol, it is very likely that there will be some XML standard for application communication over the Internet.

The DSN Operational Portal

The majority of our discussion thus far centered on mission planning and the service request process. In this section, we discuss the DSN environment during real-time operations. At present all tracks for a one week period are scheduled including specific equipment for each track (this schedule is called a "7 day schedule"). The DSN operator works from a printed version of this schedule at the station. Thus, updates, changes or problems with the schedule are done by phone or e-mail. Some tracks are longer than the view-period of one antenna. In such cases, the track must continue using an antenna in one of the other complexes around the globe. These track handovers also involve some information exchange between the two complexes, and this information exchange is also done by phone. Lastly, the status information about the track is provided to the mission by phone or e-mail.

As you can imagine, the real-time DSN operational environment is quite hectic. We believe the information exchange requirements of the DSN operational environment can be attained with an automated process that will result in a less hectic and more productive environment. To this end we have proposed an application, the DSN Operational Portal, that will enable seamless information exchange for the real-time DSN operational environment.

The proposed DSN Operational Portal is a web-enabled application for disseminating real-time operational information needed for DSN operations. The portal consists of a data store which will contain DSN operational data such as the status of ground systems, track information (both proposed and actual), mission specific equipment requirements, etc. DSN operators around the world as well as flight project personnel and possibly program managers provide information to the DSN Operational Portal and have access to the information the portal provides. Through the Internet and portal publish/subscribe and "information push" capabilities newly published information is available to all subscribers at regular (nearly immediate) updates in a secure manner. This information will manifest itself to the user, who simply connects to the DSN Operational Portal.

Portal technology is quite mature. Most people are probably familiar with several well-known portals such as AOL, Netscape Netcenter and Yahoo. Portal technology is now moving into corporations with several corporate portal offerings, e.g., Netscape Custom Netcenter and DataChannel's Rio. These products provide the publish/subscribe capabilities along with security services, configuration management and regular updates. Standards for information exchange are emerging that enable publishing of DSN operational data. The challenges for the DSN Operational Portal will be to precisely identify the information exchanged in the DSN real-time operational environment and to modify the real-time operational culture so that DSN operators and project personnel provide data in real-time and turn to the portal, rather than a telephone, fax or e-mail, to receive information.

The current DSN real-time operational environment involves a great deal of manual interaction, primarily via telephone, of DSN operators and flight project personnel. Such manual interaction and information exchange is probably inevitable for anomalous situations, but for routine operation it should not be necessary. The DSN Operational Portal will reduce current manually intensive real-time operations significantly through immediate, automated information exchange. Use of portal technology provides added value over placing DSN operational data in a database - it brings the data directly to the user. Typical database applications merely make the data available to the user if he initiates a search. The capability to personalize one's subscriptions and the portal's automated updates imply that every user can have the information they desire on their desktop any time.

We have proposed prototyping this portal with some critical functionality. The prototype would address routine data-exchange among DSN complexes including track-handover data and ground equipment information. In addition, the portal prototype will disseminate a candidate set of project information, such as changes to ground sequences or technical requirements for equipment, from the project to the DSN complexes, and selected track status information from the complexes to the project. A second functionality for the prototype addresses real-time operations during anomalous situations. The information dissemination capability of the portal is extremely valuable in handling volatile circumstances efficiently.

The first step to creating the DSN Operational Portal involves choosing a COTS portal tool. There are about ten commercial portals available and we must assess these tools with respect the DSN requirements. Since some of these tools are successfully deployed we see no reason to invest time in re-implementation. The bulk of effort toward creating the DSN Operational Portal will focus on a proof of concept of the information exchange. This project will identify in detail the information exchanged in the DSN real-time operational environment. A framework and format will be designed to enable seamless, automatic information exchange. Finally, we will develop the DSN Operational Portal application (standalone or browser-supported) and bring it to the desktop of a set of users representing TMOD and flight projects. This application will encapsulate the information exchange capabilities described in the technical summary and benefits sections and will replace the current manual information exchange for routine DSN operation.

The DSN Operational Portal is expected to reduce the manual interface time currently required for DSN and projects. Thus, personnel time is available for higher valued work. All phases of design, development and implementation will focus on the requirements of DSN operations and desires of the users - DSN and project personnel alike. Thus, we expect to create an application that provides a much-needed capability in a user-friendly manner tailored to our users and that will be readily adopted by the user community.

5. SUMMARY

We have looked at the problem of automating the operations of the Deep Space Network to support space exploration missions. The path to automation faces many technological and behavioral challenges.

We introduced XML as an up and coming technology that we strongly believe will help solve many of the technological problems. XML is a natural format for use in the heterogeneous computing environment at JPL and the DSN. Its applicability to messaging and data exchange in such environments and among applications is particularly valuable.

Section 4 of the paper describes in some detail three specific automation problems in the re-design of DSN operations. We already have a working solution to the critical problem of application integration using XML-RPC. We propose XML solutions to the problem of revising the service request process and to real-time DSN operations.

Insofar as the challenges we face in re-designing DSN operations are typical of the problems with which many organizations in aerospace are dealing, we recommend use of XML technology in this domain.

REFERENCES

[1] Jon Bosak and Tim Bray. *XML and the Second Generation Web*. Scientific American, May 1999. http://www.sciam.com/1999/0599issue/0599bosak.html.

[2] XML Specification. http://www.w3.org/TR/1998/REC-xml-19980210.

[3] XML-RPC. http://www.xml-rpc.com/.

[4] The Deep Space Network. http://deepspace.jpl.nasa.gov/dsn/.

[5] XML-RPC. http://www.xmlrpc.com/.

[6] XML-RPC Library for Java. http://helma.at/hannes/xmlrpc/.

[7] David Chappell. Simple Object Access Protocol (SOAP). Microsoft white paper. September, 1999. http://msdn.microsoft.com/workshop/xml/general/SOAP_White_Paper.asp.

[2] Michael Hammer and James Champy, *Reengineering the Corporation*, New York: Harper Business, 1993.

[3] G. Edward Bryan, "Not All Programmers Are Created Equal," *1994 IEEE Aerospace Applications Conference Proceedings*, February 5-12, 1994.

[4] G. Edward Bryan, "CP-6: Quality and Productivity Measures in the 15-Year Life Cycle of an Operating System,"" *Software Quality Journal 2, 129-144*, June 1993.

Ruth Bergman *is a Senior Computer and Information Scientist in the System and Mission Architecture section at JPL. She is working on the TMOD network simplification project among others. From 1996 through 1998, Dr. Bergman was a member of the technical staff at the MIT Lincoln Laboratory in the Advanced Systems and Sensors Group, where she developed artificial intelligence approaches for Infra-Red seeker technology in ballistic missile defense. She holds a Ph.D. in Computer Science from MIT, where she was a member of the Artificial Intelligence Laboratory and performed research in the area of* autonomous agents and machine learning.

Appendix A: Stylesheet Sample

This XSL stylesheet, when applied to the "hello world" sample in section 3, will produce the text "hello world" in red.

```
<?xml version="1.0"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/TR/WD-xsl"
    xmlns="http://www.w3.org/TR/REC-html40"
    result-ns="">

<!-- default behaviour - - - - - - - - - - - - - - - - - - - - - - - - - - --->

<xsl:template><xsl:apply-templates/></xsl:template>
<xsl:template match="textnode()"><xsl:value-of/></xsl:template>

<!-- selective behaviour - - - - - - - - - - - - - - - - - - - - - - - - - -->

<xsl:template match="GREETING">
  <DIV STYLE="font-family:Arial;font-size:15pt;color:red;"
  ><xsl:apply-templates/></DIV> <!--tag close on new line to remove
spaces-->
</xsl:template>

</xsl:stylesheet>
```